Spring Boot In 3 Weeks

Day 1: Fundamentals

Contact Info

Ken Kousen

Kousen IT, Inc.

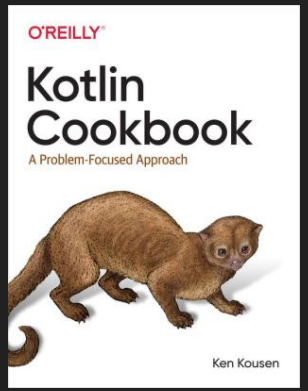
ken.kousen@kousenit.com

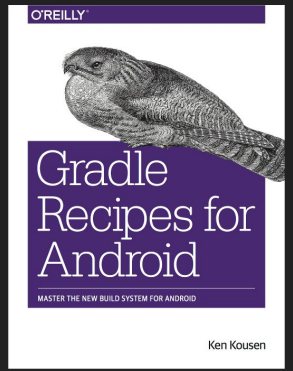
http://www.kousenit.com

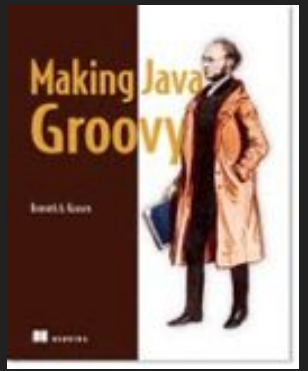
http://kousenit.org (blog)

@kenkousen (twitter)

https://kenkousen.substack.com (newsletter)







New Book

Help Your Boss Help You

https://pragprog.com/titles/kkmanage/help-your-boss-help-you/

Spring

Project infrastructure

Spring

- Lifecycle management of "beans"
  - Any POJO with getters/setters

Spring

Provides "services"

transactions, security, persistence, …

Spring

- Library of beans available
  - transaction managers
  - rest clients
  - DB connection pools
  - testing mechanisms

Spring

Need "metadata"

Tells Spring what to instantiate and configure

XML → old style

Annotations → used for standard components

JavaConfig → used for user-supplied beans

All still supported

Spring

Application Context

- Collection of managed beans
- the "lightweight" Spring container

Spring Boot

- Easy creation and configuration for Spring apps
  - Many "starters"
  - Gradle or Maven based
- Automatic configuration based on classpath
  - If you add JDBC driver, it adds DataSource bean

Spring Initializr

Website for creating new Spring (Boot) apps

http://start.spring.io

Incorporated into major IDEs

Select features you want

Download zip containing build file

Spring Boot

Application with main method created automatically

Annotated with `@SpringBootApplication`

Gradle or Maven build produces executable jar in build/libs folder

```
$ java -jar appname.jar
```

Or use gradle task `bootRun`

Spring MVC

Annotation based MVC framework

`@Controller` → controllers

`@GetMapping` → annotations for HTTP methods

`@RequestParam` and more for model parameters

`Model` interface → map for carrying data from one resource to another

Rest Client

Spring includes a class called `RestTemplate`

- Access RESTful web services
- Set HTTP methods, headers, query string, templates
- Use `RestTemplateBuilder` to create one
- Use content negotiation to return JSON or XML
- Convenient `getForObject(url, class)` method

Newer reactive client: WebClient

Logging

Spring libraries include SLF4J automatically

- Use `LoggerFactory.getLogger(... class name ...)`
- Returns an `org.slf4j.Logger` instance
- Invoke logging methods as usual

Dependency Injection

- Spring adds dependencies on request
  - Annotate field, or setter, or constructor
  - `@Autowired` → autowiring by type
  - `@Resource` (from Java EE) → autowiring by (bean) name, then by type if necessary

Testing

Spring tests automatically include special JUnit 5 extension

- `@ExtendWith(SpringExtension.class)`
- Annotate test class with `@SpringBootTest`
- Annotate tests with `@Test`
- Use normal asserts as usual

Unit Testing

Instantiate class and invoke methods

Dependencies can be mocked → Mockito is already included

Fast, but least realistic

Integration Testing

Special annotations for web integration tests

Uses Spring, but not an actual server

`@WebMvcTest(... controller class ...)`

`MockMvc` package

`MockMvcRequestBuilders`

`MockMvcRequestMatchers`

Functional Testing

Run on an actual test server

```
@SpringBootTest(webEnvironment = RANDOM)
```

- Spring chooses random port
- Deploys app
- Runs tests
- Shuts down server

Most realistic, but potentially slow

Parsing JSON

- Several options, but one is the `Jackson JSON 2` library
  - Create classes that map to JSON response
  - `restTemplate.getForObject(url, ... your class ...)`
- Maps JSON to Java objects

Component Scan

Spring detects annotated classes in the expected folders

`@Component` → Spring bean

`@Controller`, `@Service`, `@Repository` → based on `@Component`

Application properties

Two options for file name

Default folder is `src/main/resources`

- `application.properties` → standard Java properties file
- `application.yml` → YAML format

Summary for Week 1

Spring:

- Dependency injection
- Provides services
- Includes large API

Spring Boot:

- Used to create a new Spring app
- Auto-configures many beans

Great for web apps, restful web services, and more